КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інтелектуальних технологій

ЛАБОРАТОРНА РОБОТА №1

з дисципліни «Об'єктно-орієнтоване програмування»

Тема роботи: «Класи і об'єкти. Конструктори. Основи перевантаження функції.»

Варіант №10

Виконала студентка
групи АнД-21
Радоманова С.П.

Перевірила:
Москаленко Н.В.

Київ-2023

Варіант 10: Поле first - координата X, а second – координата Y точки на площині. Реалізувати метод distance(), що знаходить відстань від початку координат до точки з координатами (x;y).

Опис структури класу:
class Point - класс точки
поля: first, second - координати x, y відповідно, цілі
count - лічильник, статичне поле класу для підрахування конструкторів/ деструкторів/ конструкторів копіювання

Методи класу:
−Point() – конструктор без параметрів, проводиться ініціалізація полів нульовими значеннями;
− Point(int x, int y) – конструктор з параметрами, проводиться ініціалізація полів значеннями a та b відповідно;
– Point(const Point& other) - явно заданий конструктор копіювання, коректно копіює об’єкт
− int int getX(), setX (int coordinateX),
− int getX(), setY (int coordinateY) – методи для отримання та встановлення значення полів first та second відповідно;
− double distance() – метод для обчислення відстані до початку координат
– friend istream& operator>>(istream& is, Point& p) - функція-член класу, перевантаження оператора “>>”, дозволяє зчитувати дані в об'єкт класу Point з введеного потоку.
− friend ostream& operator<<(ostream& os, const Point& p) - функція-член класу, перевантаження оператора “<<”, дозволяє виводити дані об'єкту класу Point до виведеного потоку.
− Point operator+(int value) - метод класу, перевантаження оператора “+”, дозволяє додавати до об'єкту класу ціле.
− Point operator+(const Point& other) - метод класу, перевантаження оператора “+”, дозволяє додавати два об'єкти класу Point один до одного.
− friend Point operator+(int value, const Point& p) - функція-член класу, перевантаження оператора “+”, дозволяє додавати до цілого об'єкт класу.
− Point& operator++() - метод класу, перевантаження унарного оператора “++” префіксної форми. Вона збільшує значення об'єкта і повертає його значення.
− Point operator++(int) -метод класу, перевантаження унарного оператора “++”постфіксної форми. Вона збільшує значення об'єкта і повертає його попереднє значення.

Point
-first: int -second: int - count:static int
+Point() +Point(int x, int y) +double distance():float +operator<<(istream:istream&, const obj:Point&):istream& +operator>>(istream:istream&, obj:Point&):istream& +operator+(value: int, obj: Point): Point +operator+(obj: Point, value: int): Point +operator+(const obj: Point, const obj: Point& other): Point +operator++(obj: Point): Point +operator++(obj: Point, :int): Point

Рисунок-зображення класу мовою UML

```
Введіть кількість об'єктів класу: 3

Конструктор без параметрів спрацював!
Конструктор без параметрів спрацював!
Конструктор без параметрів спрацював!
Введіть координату X 1 об'єкту:3
Введіть координату Y 1 об'єкту:4
Введіть координату X 2 об'єкту:5
Введіть координату Y 2 об'єкту:6
Введіть координату X 3 об'єкту:7
Введіть координату Y 3 об'єкту:8
(3, 4)
Дистанція до точки: 5
(5, 6)
Дистанція до точки: 7.81025
(7, 8)
Дистанція до точки: 10.6301
Деструктор спрацював!!
Деструктор спрацював!!
Деструктор спрацював!!
Деструктор спрацював!!
Деструктор спрацював!!
```

points[1]: Point
+first = 3 +second = 4

points[2]: Point
+first = 5 +second = 6

points[3]: Point
+first = 7 +second = 8

Обрахування довжини здійснюється за допомогою теореми Піфагора:

1) $x = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = 5$
2) $x = \sqrt{5^2 + 6^2} = \sqrt{25 + 36} = \sqrt{61} = 7.81025$
3) $x = \sqrt{7^2 + 8^2} = \sqrt{49 + 64} = \sqrt{113} = 10.6301$

Після вводу кількості об’єктів в векторі ми створюємо 3 об’єкти за допомогою конструктора без параметрів, а вже потім за допомогою сеттеру записуємо значення в поля first і second. Далі за допомогою методу distance обраховуємо відстань від початку координат до точки. в кінці програми для знищення об’єктів викликаються 3 деструктори.

Текст програми:

(main.cpp)

```
#include <iostream>
#include <cmath>
#include "header.h"

using namespace std;
// Radomanova Sofiia, 10.Поле first - координата X, а second – координата Y точки на площині.
//Реалізувати метод distance(), що знаходить відстань від початку координат до
//точки з координатами(x; y).


void Function(Point p)
{
    cout << "Значення змінної count після використання конструктора копіювання:" << Point::get_count() << '\n';
```

```
    // Викликається конструктор копіювання для аргумента p
}

int Point::count = 0; // Ініціалізація статичної змінної класу


int main()
{

    //cout << "Значення змінної count до створення екземплярів класу:" << Point::get_count() << '\n';
    //
        //Завдання 1. Створення класу
        /*З головної функції створити два екземпляри розробленого класу,
    застосувавши різні форми конструкторів, вивести значення полів на екран,
        продемонструвати роботу методу відповідно до варіанту завдання.*/

    Point point1;
    //cout << "Значення змінної count після створення екземпляру класу:" << Point::get_count() << '\n';
    //point1.print(); //виклик метода прінт (використання конструктора без параметрів)
    cout << point1 << "\n";

    Point point2(2, 17);
    //cout << "Значення змінної count пісоя створення екземпляру класу:" << Point::get_count() << '\n';
    //point2.print(); //виклик метода прінт (використання конструктора з параметрами)
    cout << point2 << "\n";
    cout << '\n';
```

```
          //демонстрація роботи конструктора копіювання
     //Point point3 = point2; // Явне копіювання об'єкта
     //cout << point3 <<'\t' << point2 << '\n';
     //Point point4(point2); //Ініціалізація об'єкта через
список ініціалізаторів
     //Function(point3); // Передача об'єкта в функцію за
значенням


          //демонстрація роботи сеттеру
     cout << "Робота сеттеру для point 1: ";
     point1.setX(7);
     point1.setY(10);
     //point1.print();
     cout << point1 << "\n";

          //демонстрація роботи геттеру
     cout << "Робота геттеру для point 2: ";
     point2.getX();
     point2.getY();
     cout << point2 << "\n";
     cout << "\n";
          /*3. В кінці програми явно додати виклик деструктора
хоча б для одного
          екземпляру класу.Чи буде викликатися деструктор для
іншого екземпляру
          (відповідь обгрунтуйте) ?*/

          /*4. Описати та створити масив з екземплярів цього
класу, вивести значення
          полів на екран.Дайте відповідь на запитання : який
конструктор було
          викликано для елементів масиву ?*/
          //створення масиву об'єктів класу Point
```

```
     //Point points_1[4] = { Point(1, 1), Point(9, 3), Point(5,
8), Point(12, 2) };
     //cout << "Значення змінної count після створення
екземплярів класу в масиві:" << Point::get_count() << '\n';
     //for (int i = 0; i < 4; i++)
     //{
     //   cout << points_1[i];
     //}


          /*5. Для елементів масиву передбачити введення даних,
а також виведення
          результатів роботи методу відповідно до варіанту
завдання.*/
     int number;
     cout << "\nВведіть кількість об'єктів класу: ";
     cin >> number;
     cout << '\n';
     Point* points = new Point[number];
     //cout << "Значення змінної count після створення
екземплярів класу в векторі:" << Point::get_count() << '\n';

     for (int i = 0; i < number; i++)
     {
          cout << "Введіть координату X " << i+1 << " об'єкту:";
          int x;
          cin >> x;
          points[i].setX(x);

          cout << "Введіть координату Y " << i+1 << " об'єкту:";
          int y;
          cin >> y;
          points[i].setY(y);
```

```
    }

    for (int i = 0; i < number; i++)
    {
        cout << points[i] << '\n';
        cout << "Дистанція до точки: " << points[i].distance()
<< '\n';
    }


    /*cout << "Об'єкт + Об'єкт: " << points[0] + points[1] <<
'\n';
    cout << "Введіть число для додавання: ";
    double numberforadding;
    cin >> numberforadding;
    cout << "Об'єкт + число: " << points[0] + numberforadding
<< '\n';
    cout << "Число + об'єкт: " << numberforadding + points[0]
<< '\n';
    cout << "Префіксна форма (++point): " << ++points[0];
    Point temp = points[0]++;
    cout << "Постфіксна форма (point++): " << temp << '\n';*/


     //cout << "Значення змінної count перед викликом
деструкторів:" << Point::get_count() << '\n';

    delete[] points;
    // point1.~Point(); // явний виклик деструктора для 1 точки

    return 0;
}
```

(header)

```
#pragma once
#include <iostream>
#include <cmath>

using namespace std;

class Point
{

private:
     int first; // coordinate x
     int second; // coordinate y
     static int count; //статичний метод належить самому класу,
а не його екземплярам, тому його можна викликати без створення
об'єктів цього класу


public:
     Point()
     {
          cout << "Конструктор без параметрів спрацював!\n";
          count++;
     }

     Point(int x, int y)
     {
          this->first = x;
          this->second = y;
          cout << "Конструктор з параметрами спрацював! \n";
          count++;
     }

     // Конструктор копіювання
     Point(const Point& other)
     {
```

```
        first = other.first;
        second = other.second;
        cout << "Конструктор копіювання викликано\n";
        count++;
    }


    ~Point()
    {
        cout << "Деструктор спрацював!!\n";
        count--;
    }

    // getter for first and second coordinate
    int getX() { return first; }

    int getY() { return second; }

    static int get_count() { return count; }

    // setter for first and second coordinate
    void setX(int coordinateX) { first = coordinateX; }

    void setY(int coordinateY) { second = coordinateY; }

    void print() { cout << "X: " << first << "  Y: " << second
<< '\n'; }


    //Завдання 2. Перевантаження операцій введення/виведення

    friend ostream& operator<<(ostream& os, const Point& p) //
friend дає доступ до private поля; передаємо посилання на об'єкт
типу ostream куди буде записано результат та на об'єкт класу
Point
```

```
    {
        os << "(" << p.first << ", " << p.second << ")"; // os
- посилання на потік виводу
        return os;
    }

    friend istream& operator>>(istream& is, Point& p)
    {

        is >> p.first >> p.second;
        return is;
    }

    // Метод для обчислення відстані до початку координат
    double distance()
    {
        return sqrt(pow(first, 2) + pow(second, 2));
    }

    // Перевантаження оператора + для obj1 + obj2
    Point operator+(const Point& other)
    {
        return Point(first + other.first, second +
other.second);
    }

    // Перевантаження оператора + для obj1 + число
    Point operator+(int value)
    {
        return Point(first + value, second + value);
    }

    // Перевантаження оператора + для число + obj1
    friend Point operator+(int value, const Point& p)
    {
```

```
        return Point(p.first + value, p.second + value);
    }

    // Префіксна форма ++obj
    Point& operator++()
    {
        ++first;
        ++second;
        return *this;
    }

    // Постфіксна форма obj++
    Point operator++(int)
    {
        Point temp = *this; //  Створюємо копію поточного
об'єкта, розіменування вказівника
        first++;
        second++;
        return temp;        //  Повертаємо копію об'єкта з
початковими значеннями
    }


};
```